University of Bahrain

College of Information Technology
Department of Computer Science

ITCS241 Assembly Language Programming

Summer Semester 2008/2009
First Exam — One Hour

Q#	MARKS	
1	10	
2	15	
3	15	
TOTAL	40	

NAME: _____

ID#: _____

Q1. **[10 pts.]** Select the appropriate answer for each of the following.

(1) The largest signed 16-bit number is:

(a) 0FFFFh (b) 1000h (c) 7FFFh (d) 8FFFh

(2) Consider the following instructions:

```
mov     ax, 2A7Fh
imul    al
```

The result is stored in _____

(a) EAX and CF = 0 (b) AX and CF = 0 (c) DX:AX and CF = 0 d) AX and CF = 1

(3) When the CPU executes a single machine instruction, which order of the following operations is correct.

(a) Fetch, Execute, Decode (b) Decode, Fetch operands, Execute. (c) Decode, Fetch, Execute

(d) None of the above.

(4) Which of the following correctly defines an array of 30 dword elements, all initialized to zero.

```
A1    dword   5 dup(3 dup(2 dup(0)))
A2    dword   10 dup(3 dup (0))
A3    dword   5 dup(5 dup(0)), 5 dup(0)
A4    dword   30 dup('0')
```

(a) Only A2 and A4 (b) Only A4 (c) Only A1, A2, and A3 (d) Only A2.

(5) Which of the following instructions involves a Base-Index Displacement addressing mode.

(a) `MOV EAX, [ESI]` (b) SUB DATA[EBX+ESI], AL (c) XCHG EAX, [EBX+10]

(d) None of the above.

(6) Which of the following is not a valid assembly instruction:

(a) `MOV EAX, BX` (b) `ADD [EAX], [ECX]` (c) `INC EDX, 1` (d) All of the above.

(7) Assume you have the following data definition:

```
VALUE1    DWORD    0A23C00FBh
```

What will be moved into AX when the instruction `MOV AX,WORD PTR VALUE1+1` is executed.

(a) 003Ch (b) 00FBh (c) 3C00h (d) 0A23Ch

(8) Which of the following operators returns the number of bytes in an array.

(a) TYPE (b) SIZEOF (c) LENGTHOF (d) LABEL

(9) Which of the following is true about the LOOP instruction. Assume the label used in the loop is named L1.

(a) `if (--ECX == 0) JMP L1;`
(b) `if (--ECX != 0 ) JMP L1;`
(c) `if (ECX-- != 0 ) JMP L1;`
(d) `if (ECX-- > 0) JMP L1;`

(10) Consider the following loop:

```
        mov  ecx, 2
LL:     mov al,'A'
        call WriteChar
        loop LL
        loop LL
```

How many times the letter 'A' is displayed.

(a) 0 (b) 2 (c) infinity (d) None of the above.

Q2. Assume the following data definition.

```
.data
MyData    dword        150 dup(0)
```

1. **[9 pts.]** Write assembly instructions that store random integers in **MyData** array. The random integers should be in the range 1900 to 2009.

```
        call    Randomize
        mov     esi, 0
        mov     ecx, LENGTHOF MyData

        mov     ebx, 2009
        sub     ebx, 1900
        inc     ebx
L1:     mov     eax, ebx
        call    RandomRange
        add     eax, 1900
        mov     MyData[esi*4], eax
        inc     esi
        loop    L1
```


2. **[6 pts.]** Use **JECXZ** and **JMP** instructions to test the value stored in the last element of **MyData**. If it is divisible by 4, display the message "It is a leap year!".

```
leapMsg     BYTE    "It is a leap year!", 0
.....
        mov     eax, MyData[149*4]      ; 596
        call    WriteInt
        mov     edx, 0
        mov     ebx, 4
        div     ebx
        mov     ecx, edx
        jecxz   L2
        jmp     L3

L2:     call    Crlf
        mov     edx, OFFSET leapMsg
        call    WriteString
L3:
```

Q3. [**15 pts.**] Write assembly instructions that reads x as an integer and prints the following sequence of numbers using the LOOP instruction:

$$-x \quad +2x \quad -3x \quad +4x \quad -5x \quad +6x \quad -7x \quad +8x \quad -9x \quad +10x$$

For example, suppose the user entered $x = 2$, then your output should be:

$$-2 \quad +4 \quad -6 \quad +8 \quad -10 \quad +12 \quad -14 \quad +16 \quad -18 \quad +20$$

```
INCLUDE Irvine32.inc
.data
getMsg      BYTE "Enter x: ",0
x           DWORD ?

.code
main PROC
    mov     edx, offset getMsg
    call    WriteString
    call    ReadInt
    mov     x, eax

    mov     ebx, 1
    mov     ecx, 10
L1:
    neg     x
    mov     eax, ebx
    imul    x
    call    WriteInt
    mov     al, 9
    call    WriteChar
    inc     ebx
    loop    L1

    Call    WaitMsg
    exit
main ENDP
END main
```